


# ZIM 9.10

## **Graphical User Interface**



# What is Zim 9.10

## Zim is



**a complete framework to develop and run professional and mission critical applications by tightly integrating a lean relational database, a powerful Fourth Generation Language, an integrated development tool, the integration with outside world and client user interfaces.**



# User Interfaces

ZIM User Interfaces are a combination of the following objects defined in the Data Dictionary and maintained via ZimIDE:

**Windows:** one window per user interface to hold all objects related to this UI;

**Menus:** a particular UI may have a menu to control the actions to be performed by the application or by a section of it;

**Forms:** the screens themselves holding objects (FFS) like entry fields, labels, buttons, etc. A UI only has one form;

**Displays:** a convenient combination of forms to form a UI. Only one Display make up a particular UI;

**Form Fields (FFS):** the most basic object like entry fields, labels, buttons, etc., populating a Form.



# The UI: Definitions

**Name:** all objects have a unique name up to 18 characters;

**Navigation:** the process of moving the attention from one object to another;

**Focus:** is the location (object) where the next action will take place like losing the focus, clicking, etc.;

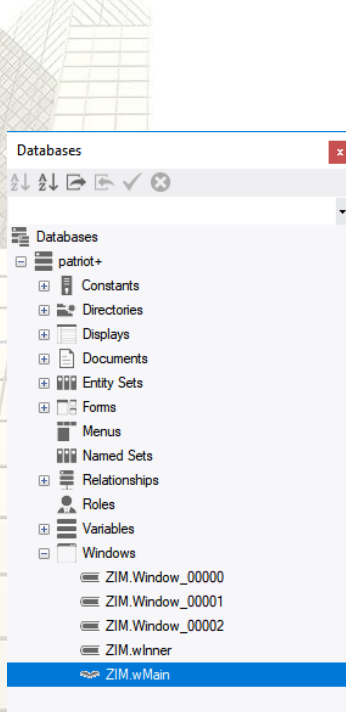
**Availability:** is the capability of the focus to interact with the user. For instance, a button may be protected, that is, the user cannot click on it; an entry field ready to accept input, that is, available for input, etc.;

**Z-Order:** the order the user navigates through all objects in the UI;

**Callback Event:** response of the object on the focus when the user interacts with the object. It can be a lose focus, click, etc.;

# Windows

The definition of the Window wMain as seen in ZimIDE (not all properties are being shown).



A screenshot of the 'Properties' panel for the ZIM.wMain window. The panel is divided into several sections: Identification, Appearance, Properties, Active Platform, and Callback Events. The Appearance section is expanded, showing various window properties.

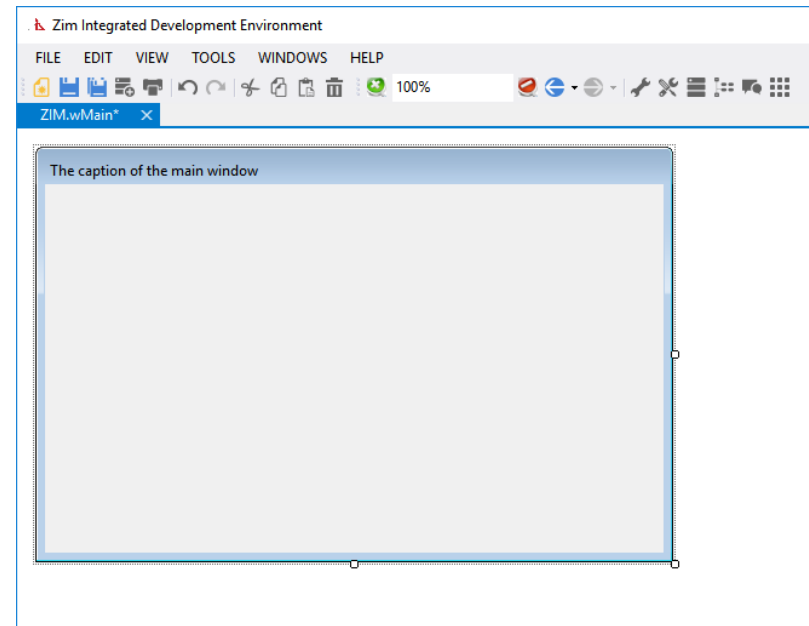
| Identification |       |
|----------------|-------|
| Window Name    | wMain |
| Directory Name | ZIM   |

| Appearance       |                           |
|------------------|---------------------------|
| Caption Bar      | True                      |
| Caption Text     | The caption of the main w |
| Clip to Parent   | False                     |
| Maximize         | True                      |
| Minimize         | True                      |
| Close Button     | True                      |
| Scroll Bars      | True                      |
| Size             | 527, 343                  |
| Position         | 0, 0                      |
| Colors           |                           |
| Icon             |                           |
| Icon Server Path | ^                         |
| Style            | Resizable                 |
| Status Bar       | False                     |
| Font             | Microsoft Sans Serif, 8pt |

| Properties      |              |
|-----------------|--------------|
| Window Tag      |              |
| Window Number   | 0            |
| Availability    | Available    |
| Topmost Setting | Not Top-most |
| Auto Size       | True         |
| Modal           | False        |
| Moveable        | True         |

| Active Platform |           |
|-----------------|-----------|
| Platform        | Desktop   |
| Resolution      | Custom    |
| Orientation     | Landscape |

| Callback Events |       |
|-----------------|-------|
| On Closed       | True  |
| Got Focus       | False |
| Lost Focus      | False |



# Windows Commands

The WINDOW OPEN command makes the referenced window available for work but not displays it.

```
WINDOW OPEN [window] [AT position] [SIZE winsize] \  
  [FOR form] [IN parent]
```

```
WINDOW OPEN wMain
```

```
WINDOW OPEN wMain AT CENTER SIZE MAXIMIZE
```

WINDOW CLOSE removes a previously opened window from the stack of opened windows and makes it inoperable.

```
WINDOW CLOSE [window]
```

```
WINDOW CLOSE wMain
```

# Windows Commands

The WINDOW ACTIVATE / DEACTIVATE commands expose and hide an already opened window.

```
WINDOW ACTIVATE [window] [EXPOSE|HIDE]
WINDOW DEACTIVATE [window]

WINDOW OPEN wMain AT TOP RIGHT
WINDOW ACTIVATE wMain
SLEEP 3
WINDOW CLOSE wMain
```

The above lines open the window wMain, activate it (display it), wait 3 seconds and then close it.

# Windows Commands

```
WINDOW MOVE [window] TO [position] [EXPOSE|HIDE]
WINDOW SIZE [window] [size-option]
WINDOW SET [NOT] CURRENT [window]
WINDOW SET CURSOR [option]
WINDOW STATUS [window]
WINDOW SET (options) [window]
```

Some options are:

[UN]AVAILABLE

BORDER color

[NOT] CLOSED

FILLCOLOR color

FONT expression

[NOT] FUNCTIONKEYS

[NOT] GOTFOCUS

LABEL expression

[NOT] LOSTFOCUS

[NOT] MESSAGES

[NOT] MODAL

PENCOLOR color

POINTSIZE expression

TABORDER BY setting



**SMARTCONE**  
LEADING THE EDGE





# Windows Events

The user interacts with the window which generate events that can be collected and processed by the Zim application.

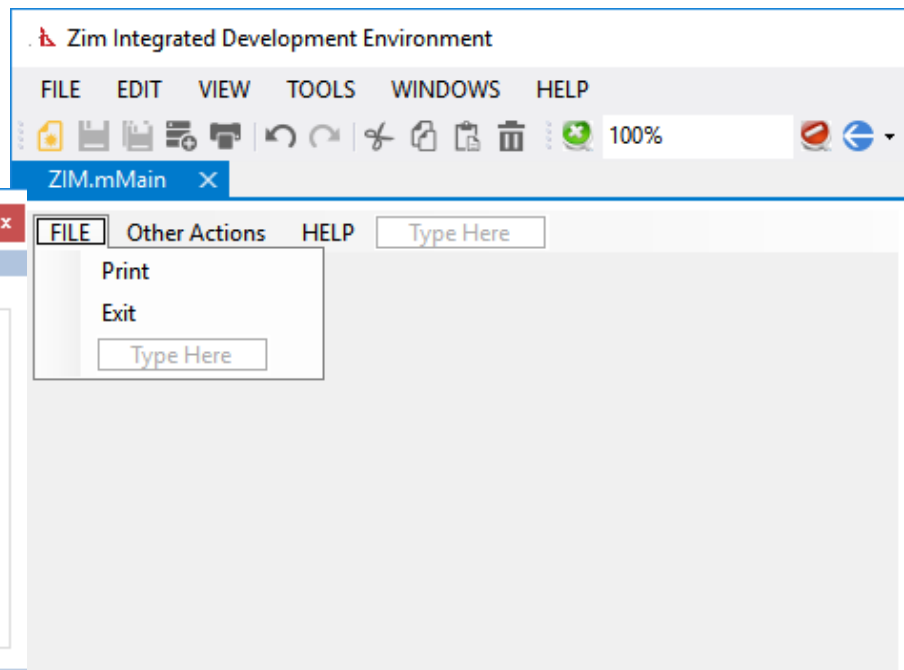
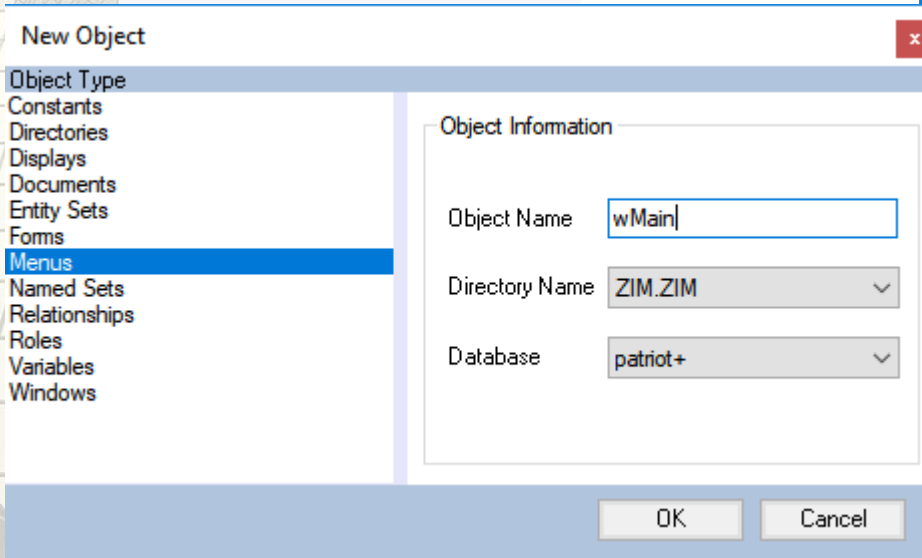
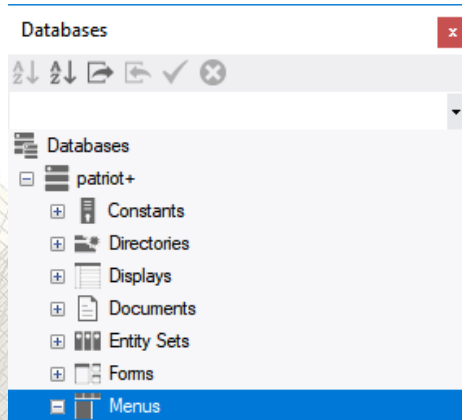
```
WINDOW OPEN wMain AT CENTER
WINDOW ACTIVATE wMain
LIST ThisWindow

WindowName WindowTag WindowNum EventName EventType ...
wMain      wMain      0          Closed      Window ...

IF ThisWindow.EventName <> "Closed"
...
...
ENDIF
WINDOW CLOSE wMain
```

# Menu

In ZimIDE, right-click on Menus, provide a new name and start placing the menu bar and menu items by simply clicking on the menu.





# Menu Commands

```
MENU OPEN menu
MENU CLOSE [menu]
MENU DISPLAY [INPUT] [PURGE] [menu]

WINDOW OPEN wMain AT CENTER
WINDOW ACTIVATE wMain
MENU OPEN mMain
MENU DISPLAY INPUT
...
...
WINDOW CLOSE wMain
```

The above lines open the window wMain, activate it (display it), open the menu mMain, display the menu and wait for the user interaction with the menu (or window).



# Menu Commands

```
MENU SET (options) object
```

```
MENU CLEAR object
```

The object is either a menu name or a menu item.

```
WINDOW OPEN wMain AT CENTER
```

```
WINDOW ACTIVATE wMain
```

```
MENU OPEN mMain
```

```
MENU SET (CHECKED LABEL "My Label") mMain.TheItem
```

```
MENU SET (SUPPRESS) mMain.AnotherItem
```

```
MENU DISPLAY INPUT
```

```
...
```

```
WINDOW CLOSE wMain
```

The names **TheItem** and **AnotherItem** were Properties given to specific menu items when creating the menu.



# Menu Events

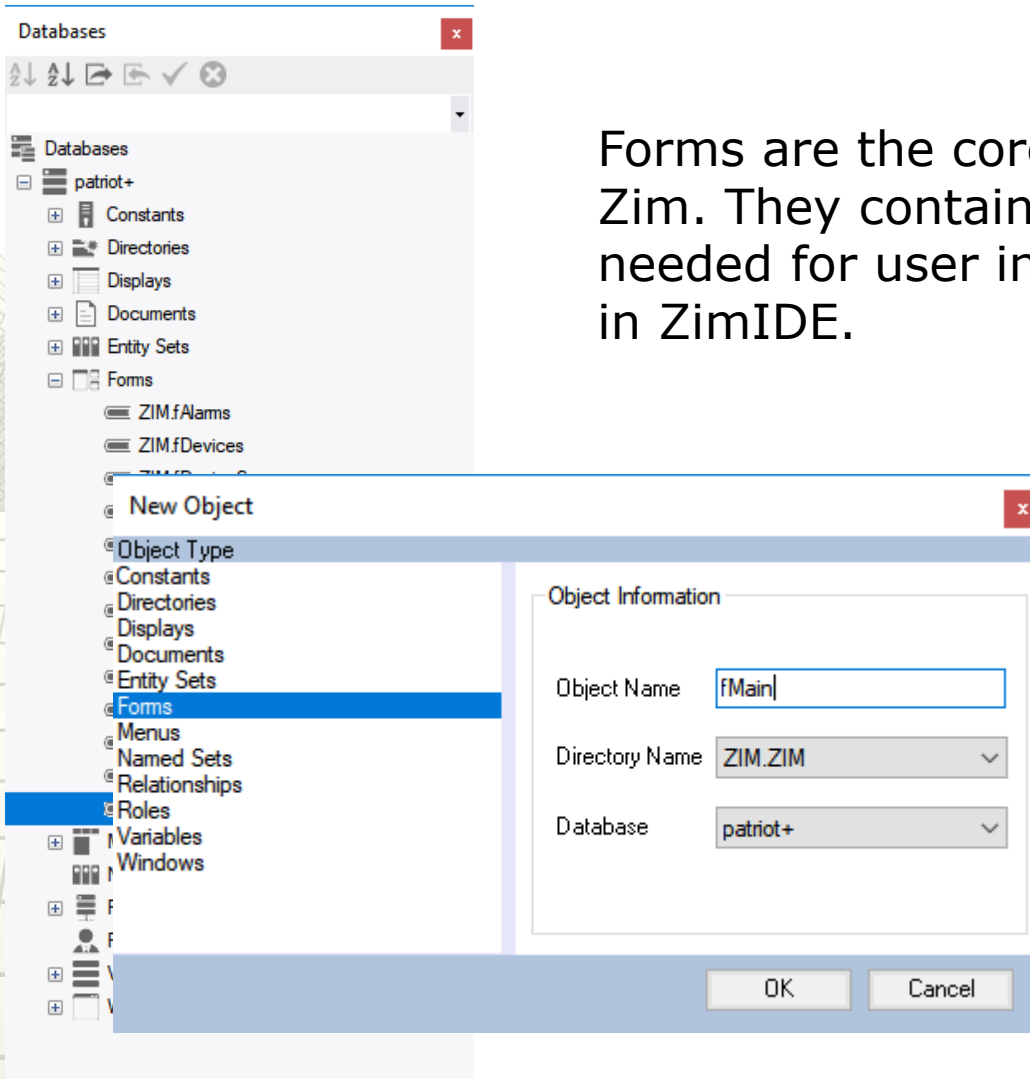
The user interaction with the menus generates events that can be trapped by the Zim application.

```
WINDOW OPEN wMain AT CENTER
WINDOW ACTIVATE wMain
MENU OPEN mMain
MENU DISPLAY INPUT
LIST ThisMenu

MenuTag MenuNum MenuChanged MenuItemTag MenuItemNum
mMain 0 0 Print 0

IF ThisMenu.MenuItemTag = "Print"
    PrintSomething()
ENDIF
WINDOW CLOSE wMain
```

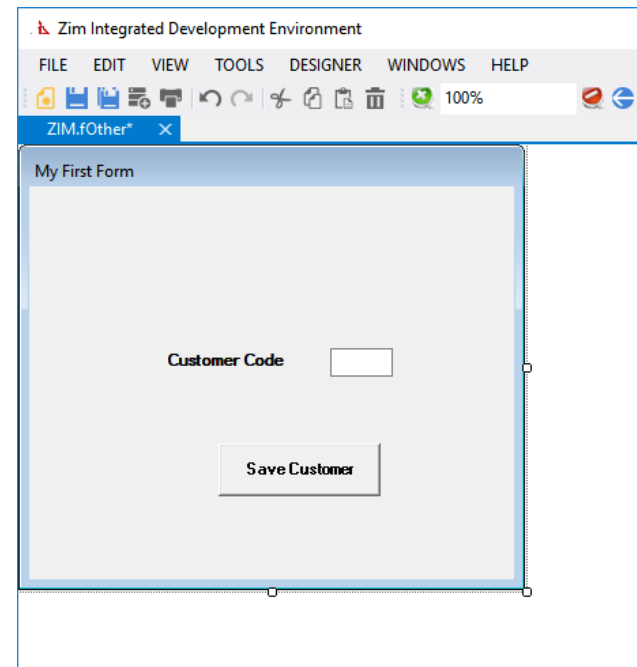
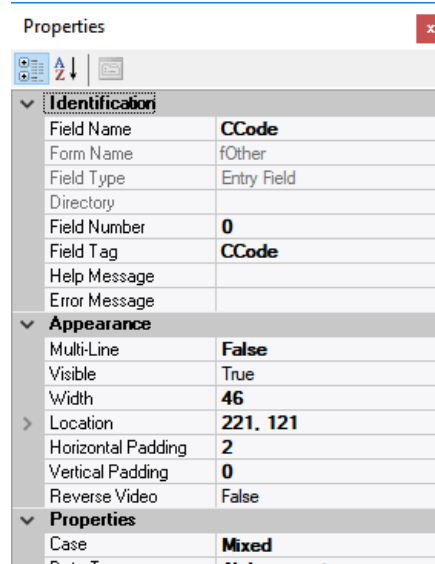
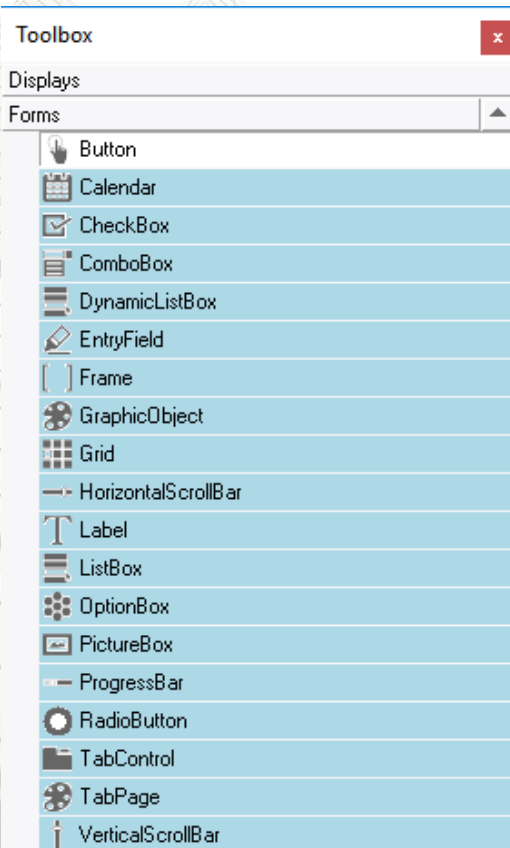
# Forms



Forms are the core of any user interface in Zim. They contain all widgets or elements needed for user interaction, all designed in ZimIDE.

# Forms

Select the widgets (Form Fields) desired from the Toolbox tab and position them in the form. From the Properties tab (only a few properties shown), configure the widget (here, the properties or attributes for the entry field CCode is shown).





# FORM Commands

```
FORM OPEN form
FORM CLOSE [form]
FORM DISPLAY [INPUT] [options] [PURGE] [object]

WINDOW OPEN wMain AT CENTER
WINDOW ACTIVATE wMain
FORM OPEN fMain
FORM DISPLAY INPUT
...
...
WINDOW CLOSE wMain
```

The above lines open the window wMain, activate it (display it), open the form fMain, display the form and wait for the user interaction with the form (or window).



# FORM SET Command

The FORM SET alters dynamically the appearance and behavior of a Zim object.

```
FORM SET (options) object
```

The object is either a form name or a form field.

% Protects the field (no input) and paints it in red.

```
FORM SET (PROTECTED FILLCOLOR RED) fMain.CCode
```

% Inhibits the double-click event.

```
FORM SET (NOT DOUBLECLICK) fMain.CCode
```

Here are some of the properties able to be used:

|                  |                 |
|------------------|-----------------|
| [NOT] ABBREVIATE | PENCOLOR color  |
| [NOT] AUTOCLEAR  | [NOT] CLICK     |
| [UN] AVAILABLE   | FILLCOLOR color |
| [NOT] BOLD       | FONT fontname   |





# The CURSOR Control

Dynamically specifies attributes that are to be applied to form fields getting the focus.

**FORM SET (options) CURSOR**

[NOT] BOLD

FILLCOLOR color

[IN]VISIBLE

[NOT] ITALIC

NORMAL

PENCOLOR color

[NOT] REVERSE

[NOT] STRIKEOUT

[NOT] SUPPRESS

[NOT] UNDERLINE

**FORM SET (FILLCOLOR RED BOLD) CURSOR**

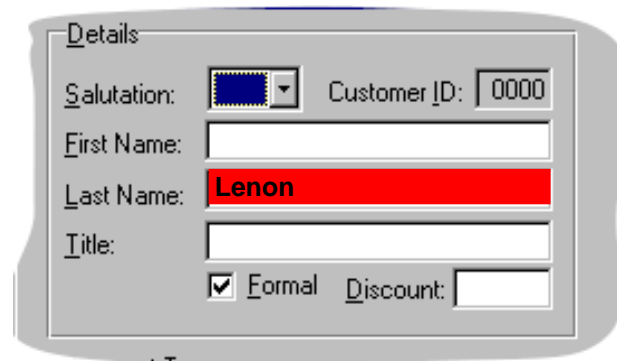
The screenshot shows a form titled "Details" with the following fields and values:

- Salutation: [dropdown menu]
- Customer ID: 0000
- First Name: John
- Last Name: **Lenon** (highlighted in red)
- Title: Director
- Formal:  Formal
- Discount: [input field]

# Working With the Focus

Sets the focus on a particular Form Field.

```
WINDOW OPEN wMain AT CENTER
WINDOW ACTIVATE wMain
FORM OPEN fMain
FORM SET (FILLCOLOR RED BOLD) fMain.LastName
FORM SET FOCUS fMain.LastName
FORM DISPLAY INPUT
...
WINDOW CLOSE wMain
```



Details

Salutation:  Customer ID: 0000

First Name:

Last Name: **Lenon**

Title:

Formal Discount:



# FORM Events

The user interaction with forms can be detected within a Zim application via the special structure **ThisForm**.

```
WINDOW OPEN wMain AT CENTER
WINDOW ACTIVATE wMain
FORM OPEN fMain
FORM DISPLAY INPUT
IF ThisForm.FormTag = "fMain"
    DoSomething()
ENDIF
WINDOW CLOSE wMain
```



# DISPLAYS

Forms can be grouped and treated as a unit within Zim objects called Displays. All Zim commands and events dealing with forms are also valid and extended to displays.

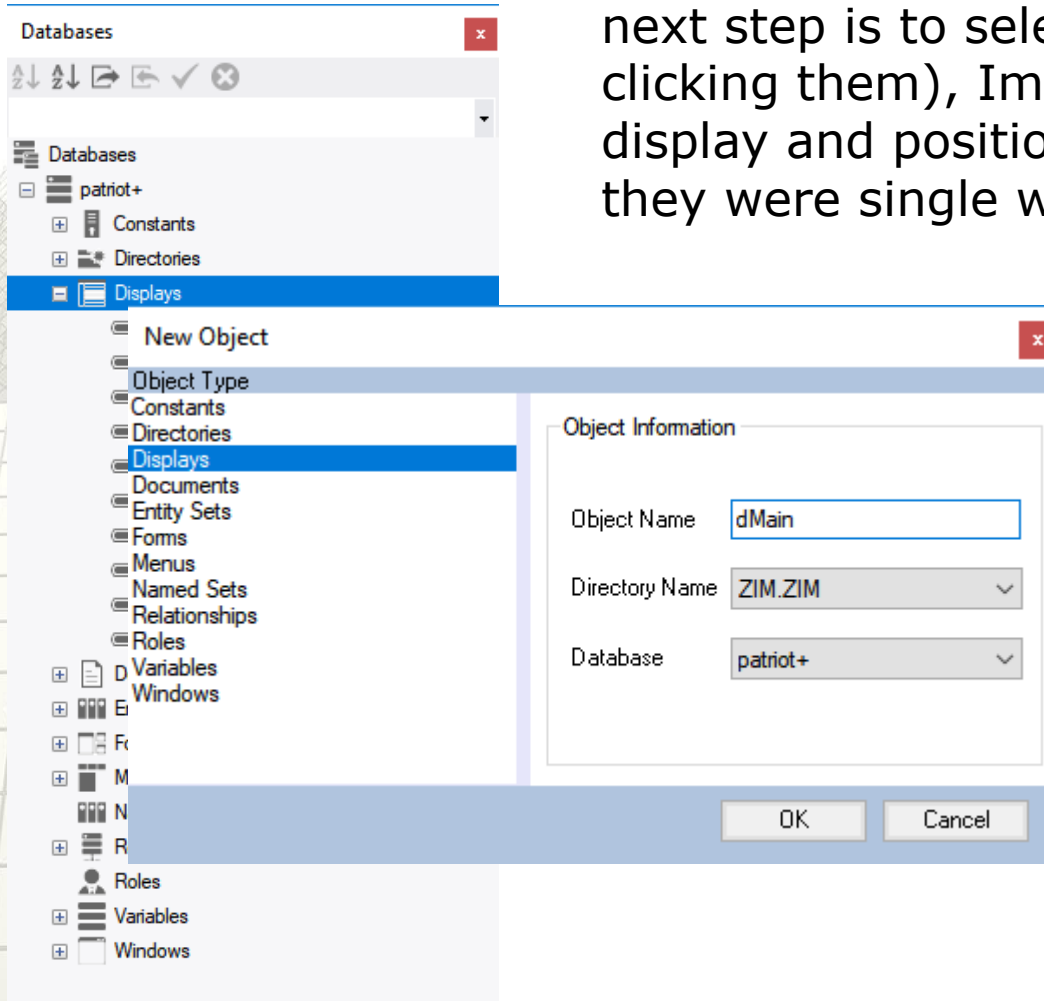
All objects within a display are accessed via their own forms. In particular, the FORM OPEN / CLOSE commands refer to a display and not any of its enclosed forms.

The same form can be defined in many different displays (for example, all screens presenting a common footer).

Another special property of displays is that a form can be repeated down and across forming a “grid” of objects that can be accessed via the \$Subscript system variable.

# DISPLAYS

Displays are created in ZimIDE. The next step is to select Forms (by right-clicking them), Import them into the display and positioning them as though they were single widgets.





# Events - Overview

In Zim, Windows, Menus and Forms may produce special responses called events (or callback events) feasible to be trapped within a Zim application by the special structures ThisWindow, ThisMenu and ThisForm.

These structures contain detailed information about the object generating the event along with the event itself.

However, there is another structure that is generic for all objects: **Event** which does not provide detailed information but generic enough to keep track of the event regardless of the originating object type.

In fact, rarely there is a need to access the specific structures.



# Events - Overview

```
WINDOW OPEN wMain AT CENTER
WINDOW ACTIVATE wMain
FORM OPEN fMain
FORM DISPLAY INPUT
LIST Event

EventName EventType EventTag      WindowTag FormTag FieldTag MenuTag ...
Click          FormField bMainProfiles wMain      fMain      bMainProfiles ...

% It does not matter where Click is coming from.
IF Event.EventName = "Click"
    DoSomething()
ENDIF
WINDOW CLOSE wMain
```





# Accelerator Keys

```
WINDOW SET [ADD | NOT] ACCELERATOR [keynames]
```

Accelerator Keys are mouse or keystrokes causing an event to be generated once set to do it.

In the example below, every time a F1, ESCAPE or ENTER are pressed, it generates an event.

```
WINDOW OPEN wMain AT CENTER
WINDOW ACTIVATE wMain
WINDOW SET ACCELERATOR F1 ESCAPE ENTER
FORM OPEN fMain
FORM DISPLAY INPUT
%If pressed ESCAPE, exit the application immediately.
IF Event.EventName = "ESCAPE"
    BYE
ENDIF
WINDOW CLOSE wMain
```

# Timing Out the Input

Controls how long a FORM INPUT or a MENU INPUT will wait for the user interaction.

```
WINDOW OPEN wMain AT CENTER
WINDOW ACTIVATE wMain
WINDOW SET ACCELERATOR F1 ESCAPE ENTER
FORM OPEN fMain
SET INPUT TIMEOUT 10
FORM DISPLAY INPUT
% Sorry! After 10 seconds, chance lost!
IF Event.EventName = "TIMEOUT"
    NextCandidate()
ENDIF
WINDOW CLOSE wMain
```



# ZIM 9.10

## **Graphical User Interface**